STAT 672 Final Project: Stochastic Gradient Descent

Tom Wallace

April 14, 2018

1 Introduction

1.1 Organization

This paper is divided into four sections. The remainder of this **Introduction** section gives intuitive motivation for stochastic gradient descent (SGD). The **Method and Theory** section more rigorously presents the mathematics of SGD and some of its notable properties. The **Applications** sections highlights the real-world settings and uses of SGD, including a case study data analysis. The **Conclusion** section summarizes overall findings.

1.2 Motivation

Optimization is fundamental to statistical modeling. The chief task of statistical modeling is to characterize the relationship between explanatory variables and an outcome variable, and the chief method for doing so is to estimate values for coefficients that best relate each explanatory variable to the outcome variable. The term "best" implies picking coefficient values that maximize some measure of goodness (e.g. likelihood) or minimize some measure of badness (e.g. loss function). Mathematical optimization is the typical route to achieving such minimization or maximization. Two important considerations for optimization are parametric assumptions and computational complexity. SGD, an optimization technique, is particularly motivated by these considerations.

1.2.1 Parametric vs. non-parametric

Assuming that the outcome variable follows a particular statistical distribution aids the computation of optimal coefficients. For example, assumptions in ordinary least squares (OLS) regression—assumptions that readers almost certainly are familiar with and so will not be repeated here—allow a closed form solution. The optimization problem of choosing coefficient $\hat{\boldsymbol{\beta}}$ that minimize squared error is solved by $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$ (where $\boldsymbol{X}$ is the feature matrix and $\boldsymbol{Y}$ the outcome variable).

Even if a parametric model does not have a closed-form solution, the parametric assumption allows some useful optimization techniques. Consider logistic regression. The maximum likelihood estimator (MLE) approach for estimating coefficients leads to a system of D equations. This system of equations typically is numerically solved using the iterative Newton-Raphson algorithm:

$$\hat{\boldsymbol{\beta}}_{n+1} = \hat{\boldsymbol{\beta}}_n - \boldsymbol{H}^{-1}(\hat{\boldsymbol{\beta}}_n)\boldsymbol{J}(\hat{\boldsymbol{\beta}}_n)$$

$\boldsymbol{J}$ is the Jacobian (the first derivative of the log-likelihood function l with respect to each w_j) and $\boldsymbol{H}$ is the Hessian (the second derivative of l with respect to $w_j, w_{j'}$). The practicality of Newton-Raphson thus depends on whether it is convenient to find $\boldsymbol{J}$ and $\boldsymbol{H}$. It is convenient for logistic regression because parametric and independent-and-identically-distributed (IID) assumptions mean l is a simple sum of the log probability distribution function (PDF, in this case binomial) for each observation. We “know” (assume) the form of this PDF and so are confident that the second derivative exists and is not too onerous to calculate. In non-parametric settings, we often cannot be so certain and face the possibility of $\boldsymbol{H}$ being non-existent or cumbersome.

The need to conduct optimization in non-parametric settings is a chief motivation for gradient descent (GD), of which SGD is a variant. In non-parametric settings—most notably supervised and unsupervised statistical learning, in which we again seek to find optimal coefficients to relate input variables to output variables for the purposes of classification or regression—there typically is no closed form solution for the coefficients. It also may not be convenient to find and evaluate the Hessian, making Newton-Raphson undesirable. SGD does not require any parametric assumptions. In its most basic form, SGD only requires finding the gradient (though some extensions do need the Hessian or an approximation to it). SGD thus is well-suited for non-parametric settings.

1.2.2 Computational Complexity

How an optimization technique scales with sample size n is another important consideration. It is little comfort if a method reaches the correct solution but requires an excessive amount of time to do so. “Plain” or “batch” GD requires evaluating the gradient for every single observation, every single iteration, until the algorithm converges. For example, for a dataset of $n = 10^6$ that required 25 iterations to converge, batch GD would require evaluating the gradient 25×10^6 times. This scaling with n can cause untenably long computation time.

SGD alleviates these computational difficulties by requiring the gradient to be evaluated for only a single randomly chosen observation per iteration. This approach means convergence is “noisier” and hence requires more iterations to converge, but each iteration is less complex to compute and so can be done faster. SGD thus scales much more favorably with n than GD, and so is particularly useful for large-n applications such as machine learning and big data problems.

2 Method and Theory

2.1 Basic setup

Loosely following the set-up of Bottou 2010, consider a typical supervised classification problem. We have feature matrix $\boldsymbol{X}_{n\times D, n\in\mathbb{R}^n, D\in\mathbb{R}^D}$ (corresponding to n observations and D features) and labels $\boldsymbol{Y}_{n\times 1}, Y_i \in \{-1, 1\}$. The goal is to predict a particular observation's label Y_i using that observation's features $\boldsymbol{X}_i$. We have a hypothesis class $\mathcal{F}$ consisting of various functions $f_{\boldsymbol{w}}(\boldsymbol{X}_i) \in \mathcal{F}$ parametrized by weight vector $\boldsymbol{w}$. We have loss function $L(Y_i, f_{\boldsymbol{w}}(\boldsymbol{X}_i))$ that expresses the cost of mis-classification. Assume for now that L is convex. We will consider the optimal function $\hat{f}_{\boldsymbol{w}}(\boldsymbol{X}_i) \in F$ that which minimizes empirical risk over all observations: $\frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_{\boldsymbol{w}}(\boldsymbol{X}_i))$. Denote $\hat{\boldsymbol{w}}$ the weight coefficients of this optimal function. We thus have:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_{\boldsymbol{w}}(\boldsymbol{X}_i)) \tag{1}$$

Because L is convex, basic calculus tells us that there is one critical point, it is the global minimum, and it is located where the gradient of the loss function with respect to $\boldsymbol{w}$, $\nabla_{\boldsymbol{w}}$, is zero. GD is an iterative algorithm to numerically approximate this point:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{w}} L(Y_i, f_{\boldsymbol{w}_t}(\boldsymbol{X}_i)) \tag{2}$$

t refers to iterations. γ is a parameter controlling the step size (also called "learning rate"). Assume for now that γ is fixed. The algorithm stops when the gradient is arbitarily close to zero; or, put differently, when the difference between this iteration's estimate and the last iteration's estimate is arbitrarily small.

$$\boldsymbol{w}_t - \boldsymbol{w}_{t-1} \leq \epsilon \tag{3}$$

Note that the batch GD algorithm presented in (2) requires evaluating the gradient for every single observation i. In a large-n dataset, this can be computationally infeasible. SGD's innovation is to instead evaluate only a single randomly chosen observation i at each iteration t, greatly lightening the computational load.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\nabla_{\boldsymbol{w}} L(Y_i, f_{\boldsymbol{w}_t}(\boldsymbol{X}_i)) \tag{4}$$

Having outlined the basic form of SGD, we now examine some of its key properties.

2.2 Key Properties

2.2.1 Correctness

2.2.2 Speed

2.3 Extensions

The basic SGD algorithm has been extended in different ways. The popularity of the algorithm disallows a comprehensive or detailed treatment of all development. This sub-section covers some of the more interesting extensions.

Bottou 2012 Boyd and Vandenberghe 2004 Dal Pozzolo et al. 2015

2.3.1 Step Size (Learning Rate)

Shalev-Shwartz et al. 2011

2.3.2 Momentum

Polyak and Juditsky 1992 Nesterov 1983

2.3.3 Averaging

2.3.4 Predictive Variance Reduction

2.3.5 Parallelization

SGD is commonly used in large-n, computationally demanding applications. Thus, even though SGD is a computational improvement over batch GD, there has been interest in whether SGD can be made even faster by parallelizing it. Zinkevich et al. 2010 present novel algorithms for doing so. The actual algorithms are strikingly simple; their proof is highly technical and omitted here.

The parallelization technique essentially is averaging. In line with previous notation, suppose we have fixed learning rate γ,

Advantages

3 Applications

3.1 SGD and Statistical Learning

4 Conclusion

References

[1] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[2] Léon Bottou. "Stochastic gradient descent tricks". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Andrea Dal Pozzolo et al. "Calibrating probability with undersampling for unbalanced classification". In: *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE. 2015, pp. 159–166.

[5] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate O(1/sqr(k))". In: *Soviet Mathematics Doklady* 27 (1983), pp. 372–376.

[6] Boris T Polyak and Anatoli B Juditsky. "Acceleration of stochastic approximation by averaging". In: *SIAM Journal on Control and Optimization* 30.4 (1992), pp. 838–855.

[7] Shai Shalev-Shwartz et al. "Pegasos: Primal estimated sub-gradient solver for svm". In: *Mathematical programming* 127.1 (2011), pp. 3–30.

[8] Martin Zinkevich et al. "Parallelized stochastic gradient descent". In: *Advances in neural information processing systems*. 2010, pp. 2595–2603.